The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method of executing a native method in a Java virtual machine comprising:

in the Java virtual machine, determining whether [[a]] the native method is to be handled by a first native interface or one of a plurality of other native interfaces;

if the native method is to be handled by the first native interface, invoking the method and enabling the native method to access an internal state of the Java virtual machine;

~~executing the method in the Java virtual machine;~~ and

adjusting the state of the Java virtual machine based on execution of the native method ~~whereby transition between an interpreter loop and the native method via the first native interface is minimized~~.

2. (Original) A method as recited in claim 1 further comprising classifying one or more native methods so that the one or more native methods qualify for being handled by the first native interface.

3. (Original) A method as in claim 1 further comprising:

eliminating a need to push a Java stack frame onto a Java stack;

eliminating a need to marshal one or more arguments and a method result from the Java stack to a C stack;

eliminating a need to marshal the method result from the C stack to the Java stack; and

eliminating a need to pop the Java stack frame from the Java stack.

4. (Currently amended) A method as recited in claim 1 wherein determining whether [[a]] the native method is to be handled by a first native interface or one of a plurality of other native interfaces further comprises examining a method block of the native method to determine a method type.

5. (Original) A method as recited in claim 1 further comprising:

obtaining a function pointer from a method block;

invoking the native method function; and

passing to the native method function one or more arguments that allow access to a Java virtual machine state to be used by the native method without making callbacks to the Java virtual machine.

6. (Original) A method as recited in claim 5 further comprising:

passing to the native method a pointer to arguments on a Java stack; and

passing to the native method a pointer to a method block pointer, such that a new method block pointer can be returned to the interpreter loop.

7. (Original) A method as recited in claim 1 wherein executing the method in an interpreter loop further comprises:

pushing a transition frame corresponding to a particular method onto a first stack in the Java virtual machine;

the native method pushing a plurality of arguments associated with the transition frame onto the first stack; and

returning a result code to the interpreter loop.

8. (Original) A method as recited in claim 7 wherein the first stack is a Java stack and the result code indicates that a new transition frame has been pushed on the Java stack.

9. (Original) A method as recited in claim 1 wherein adjusting the state of the JVM further comprises:

storing a result from the native method on a Java stack; and

modifying a Java stack pointer based on a return code.

10. (Cancelled)

11. (Currently Amended) A system for executing a native method in a Java virtual machine, comprising:

a processor; and

a computer-readable medium storing a program for execution by the processor, the program comprising:

computer code in the Java virtual machine that determines whether [[a]] the native method is to be handled by a first native interface or one of a plurality of other native interfaces;

computer code that invokes the native method and enables the native method to access a state of the Java virtual machine, if the native method is to be handled by the first native interface; and

~~computer code that executes the method in the Java Virtual Machine; and~~

computer code that adjusts the state of the Java Virtual Machine based on execution of the native method ~~whereby transition between an interpreter loop and the native method via the first native interface is minimized~~.

12. (Currently Amended) A computer-readable medium containing programmed instructions of a Java virtual machine arranged to execute a native method in [[a]] the Java virtual machine, the computer-readable medium including programmed instructions for:

determining whether [[a]] the native method is to be handled by a first native interface or one of a plurality of other native interfaces;

if the native method is to be handled by the first native interface, invoking the native method and enabling the native method to access a state of the Java virtual machine;

~~executing the method in the Java Virtual Machine~~; and

adjusting the state of the ~~JVM~~ Java virtual machine based on execution of the method ~~whereby transition between an interpreter loop and the native method via the first native interface is minimized~~.